

Gestión de pacientes en una clínica

En la clínica "La Fuerza" se requiere desarrollar un sistema de gestión de pacientes. El sistema debe gestionar la información de los pacientes atendidos en el día. Para cada paciente, se almacenará:

- Número de historia clínica (un número entero).
- Nombre del paciente (una cadena de texto).
- Edad del paciente (un número entero).
- Diagnóstico (una cadena de texto).
- Cantidad de días de internación (un número entero).

La información de todos los pacientes debe almacenarse en un array bidimensional, donde cada fila representará un paciente, y las columnas contendrán los datos mencionados arriba. Recordar que el array debe comenzar vacío.

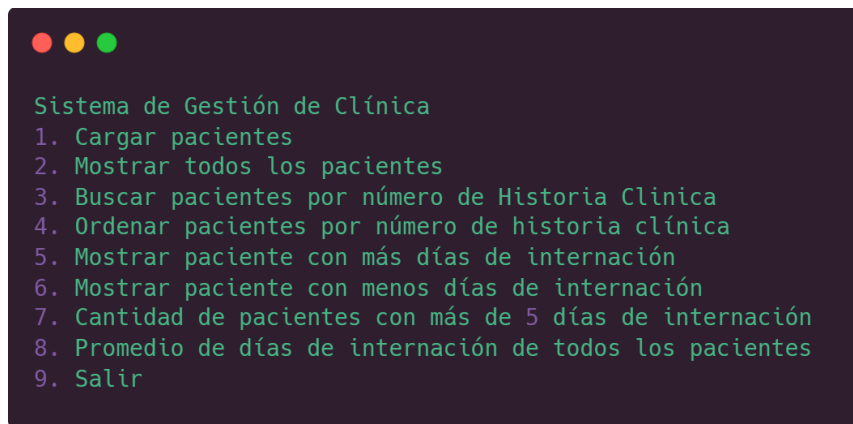
Presentamos un ejemplo de cómo debería verse:

```
pacientes = [  
    [1234, "Jorge Gomez", 54, "Fiebre", 2],  
    [1235, "Ana Perez", 25, "Hipotiroidismo", 5],  
    [1236, "Carlos Lopez", 30, "Neumonia", 5],  
    [1237, "Ludmila Gonzalez", 22, "Celulitis orbitaria", 10],  
]
```

Requerimientos:

1. Interfaz del programa:
 - El sistema debe mostrar un menú interactivo para que el usuario pueda elegir entre las diferentes opciones del sistema (cargar pacientes, buscar paciente por Historia Clínica, determinar paciente con más/menos días de internación, ordenar pacientes por número de historia clínica, salir del sistema, etc.).
 - El menú debe estar dentro de un bucle que permita al usuario realizar múltiples operaciones hasta que decida salir.

Ejemplo de cómo podría verse el menú

A screenshot of a terminal window with a dark background and light green text. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The text in the terminal is as follows:

```
Sistema de Gestión de Clínica
1. Cargar pacientes
2. Mostrar todos los pacientes
3. Buscar pacientes por número de Historia Clínica
4. Ordenar pacientes por número de historia clínica
5. Mostrar paciente con más días de internación
6. Mostrar paciente con menos días de internación
7. Cantidad de pacientes con más de 5 días de internación
8. Promedio de días de internación de todos los pacientes
9. Salir
```

2. Cargar pacientes:
 - Permitir al usuario ingresar los datos de los pacientes, almacenando la información en una lista anidada (arreglo bidimensional), como se muestra en la imagen de arriba. La cantidad de pacientes a ingresar debe ser determinada por el usuario.
3. Mostrar la lista de pacientes:
 - Imprimir en pantalla todos los datos de los pacientes almacenados en el arreglo bidimensional, mostrando cada fila como un paciente.
4. Búsqueda de pacientes:
 - Implementar una función que, dado el número de historia clínica de un paciente, busque en la lista y muestre todos los datos de dicho paciente (o un mensaje indicando que no se encontró al paciente).
5. Ordenamiento de pacientes:
 - Implementar una función que permita ordenar la lista de pacientes por el número de Historia Clínica en forma ascendente. Se podrá utilizar cualquier algoritmo de ordenamiento.
6. Determinar el paciente con mayor cantidad de días de internación:
 - Implementar una función que calcule e imprima el paciente con más días de internación, mostrando sus datos completos.

7. Determinar el paciente con menor cantidad de días de internación:
 - Implementar una función que calcule e imprima el paciente con menos días de internación, mostrando sus datos completos.
8. Cantidad de pacientes con días de internación mayor a 5 días.
 - Implementar una función que recorra la lista de pacientes y cuente cuántos pacientes tienen más de 5 días de internación.
9. Promedio de días de internación de todos los pacientes.
 - Implementar una función que calcule el promedio de días de internación de todos los pacientes registrados.

Importante:

- Si el usuario selecciona cualquier opción sin que existan pacientes registrados en el sistema, aparecerá un mensaje en pantalla notificando que no hay pacientes registrados para la operación solicitada.

Sugerencias:

- Se recomienda la validación de los datos ingresados por el usuario.
- Organizar el código en funciones y, si es posible, en módulos independientes. Utilizar `import` para incorporar funcionalidades de otros módulos cuando sea necesario.
- Se aconseja incluir documentación de funciones, sugerencia de tipos y comentarios explicativos donde lo consideren necesario.
- Utilizar las estructuras que vimos hasta este momento de la cursada.
- Utilizar correctamente arreglos bidimensionales para almacenar la información.
- Proporcionar salidas claras para el usuario. La presentación de los datos debe ser comprensible y legible. Se recomienda utilizar f-string.

Entrega del programa

- La entrega se deberá realizar mediante el link al repositorio de GitHub.